

1 - User Profiles and Segments

- 1) Dashboard value keeps **updating number every 5 seconds and completely updates in 5 minutes**
- 2) When first installed app or website, the user is anonymous. Clevertap creates a **clevertap id** for it. Android - google ad id is used to create its own id. IOS - id epic
- 3) With GDPR- clevertap doesn't have access to these from the device. In such cases, clevertap **creates random ids for user profiles**
- 4) For website, a cookie is drawn up known as clevertap id again
- 5) When a **user registers, that's when clevertap user profile data can be sent**
- 6) **Identity and email id = primary identifier** by default can be changed according to customer or use case
- 7) Only clevertap id/identity can also be used as the primary identifier.
- 8) Green bell icon means push notifications can be sent.
- 9) When app uninstalled, the **user can't be reached via push notification as push token is deleted.**
- 10) Subscriptions can be managed via UI itself
- 11) **Profile properties (256) can be stored** - can be any important info about user, like age, gender, customer ID, timeZone etc. needs to be sent.
- 12) Scenario-
 - a) If a user has a second device which is not initially logged in. But they have registered already for the app on their primary device, will a second ClevertapId generated as on the first login, clevertap can't check user info?
 - b) Clevertap actually merges the clevertap identity to merge the two devices together
 - c) The identity is already set to identify the user from any device, in that case when the user finally logs in from the second device then, the profiles from two devices are merged back to represent a single profile.
 - d) Typically the customer gives a customer id or an identifier like that to identify their users from any device, which will be used to merge profiles
- 13) Two types of location tracking. - **city location and latitude/longitude.**
- 14) City location - via reverse IP lookup.
- 15) Lat/long - via the app.
- 16) Customer needs to be GDPR compliant for location tracking.

- 17) Clevertap **has two types of tracking, one on by own (System events) and one by custom events**
- 18) On schema-> events we can see what is being tracked via system
- 19) We need not do any additional coding to define **system events, they are automatically tracked**
- 20) Custom events need to be defined on their own
- 21) **512 events custom events can be defined** on clevertap. Each event can have **256 properties on their own**
- 22) E.g payment custom event can have - added, charged, searched, reachable etc
- 23) There is **role-based access control on clevertap** dashboard which is completely controlled by business
- 24) **Groups of users are called segments** which contain a particular collection of users bases on some criteria. This is the second part of profiles and is used to engage a large group of users collectively
- 25) Segments can be created on the segments tab.
- 26) Primarily there are two types of segments - **Live user segments** and **Past behaviour segments**
- 27) Three icons in past behaviour segment lead to the same type of segment build up screen.
- 28) There are three parts in the segment build-up - **Interests, behaviours and common properties.**
- 29) These run queries in clevertap to find the type of users with these properties.
- 30) The live user segment helps in interacting with users currently doing a task.
- 31) The most important in these are single action and inaction within time.
- 32) Inaction within time can be understood with an example - if a user is doing a transaction and for some reason, he/she stops and does not create the transaction so that the user can be tracked.
- 33) Live user is updated every 5 seconds. Past behaviour segments change every midnight.
- 34) In the custom list live user segment, the id or the email id must be mapped.

2 - Analytics tools

- 1) All created segments are available for use in the analytics section.
- 2) Event is a simple dashboard, it helps in getting a detailed analysis of already tracked events.
- 3) The results are displayed on the right side and show the output to the criteria of the event.

- 4) By clicking on view events we can get more info about the event.
- 5) There are 6 tabs in the events dashboard - **Quick View, Trend & Properties, Session, Geography, Technographics and People.**
- 6) Quick View, just gives a brief info about the event and what all occurred in it.
- 7) Trends show graphs telling how the event has been performing daily, monthly or weekly. It also shows what event property is trending and how it has been performing.
- 8) The session shows the best time for the event and compares it to the other session times.
- 9) Technographics shows app version, os, device details etc. It can be used to track which version or device performs the best.
- 10) People get a sample user data to show to understand the type of users interacting with the event, along with demographics of the event
- 11) Funnels** are used to understand drop ups or conversions at each stage of the user lifecycle.
- 12) E.g How many users viewed the product, how many added that to cart and how many actually did the transaction. We can also add criteria for how much time this funnel takes to complete its cycle or the properties like specific products we want to track.
- 13) We can get more information about a funnel by simply clicking on view funnel.
- 14) A graph will show each stage of the lifecycle.
- 15) The grey area above a graph is clickable and we can create segments or campaign for these users. The grey area shows the people dropped from the previous lifecycle.
- 16) If we chose the hold option, we can create a funnel which holds a certain property for the entire funnel lifecycle. E.g if a user views shoes and we want to see the funnel for just that shoe, the hold category option will give us a contextual result for that.
- 17) Further, we can do a breakdown based on an event or user property on a particular step.
- 18) This will give a list of all categories like shoes, hoodies etc that will show up as separate graphs telling details about funnel conversions etc.
- 19) Cohorts** are useful in understanding the retention or repeat transaction behaviour.
- 20) This can give us a weighted average for every day to see how many users came back on our app.
- 21) Allow subsequent reentries option allows us to track a user twice if he has re-entered the app but on separate days.
- 22) Trends** is a powerful tool to get deeper insights into our own data. It can analyse individual property trends, compare it with different events and visualise it in form of a graph.

- 23) The trends can be plotted for **Event Count, Unique Users, Events per user, Active %, Properties, Formula.**
- 24) In the setting section of the dashboard, we can go to schema, events and then qualifying event to set the Active % of users criteria to show up in trends.
- 25) **Pivots** help us in getting insights about the data. Typically we create pivot tables in excel to get very specific insights about our data.
- 26) Clevertap allows the pivot table on the dashboard itself.
- 27) **Flows** give us a 360 view of all the possible funnels. Funnels in themselves are one dimensional as they give just conversions or dropouts. Flows give us analysis about an entire lifecycle telling us better what happened
- 28) There are two types of flows. 1) **Event Flows** 2) **Content Flows.**
- 29) In **Event Flow**, We can change events in event flows to see all possible paths users took in an event. E.g visualise what users did after and before performing an event
- 30) We read the event flow chart inside to outside, inside being the most important event, and outside the least in importance.
- 31) If there is an end event after app launch, it means users did nothing after launching the app.
- 32) This is just an observational tool and we can't interact with data. Only actionable tools are events, cohorts and funnels.
- 33) In **Content Flow**, unlike event flow where we take only an event into consideration, we take an event property into consideration.
- 34) **Real Impact.** Before understanding the real impact we need to understand control groups. Once we create a control group, these users cannot receive any notifications or campaigns from us. By this way, we can understand the performance of our control group who aren't targeted by campaigns and events, to those of targeted groups which receive all these events and campaigns
- 35) With Real Impact we want to understand the overall efficiency of our campaigns.
- 36) This is measured by **Sticky Quotient, Retention, Conversion, Revenue per user, RFM Segments, Recency**
- 37) **The sticky quotient** is simply **Daily Active Users/Monthly Active Users**. This helps us understand that how many daily active users become are monthly active users. Similarly, we can track these for all other tabs.
- 38) We can choose to remove the control group for very important campaigns.
- 39) **RFM** stands for **Recency and frequency analysis**. This helps us to create an auto segmentation of our users on clevertap and understand which users are the most powerful users of the customer's app.
- 40) Three important parameters are considered, **Recency, Frequency and monetary analysis.**
- 41) Different events can be chosen for recency and frequency and monetary.

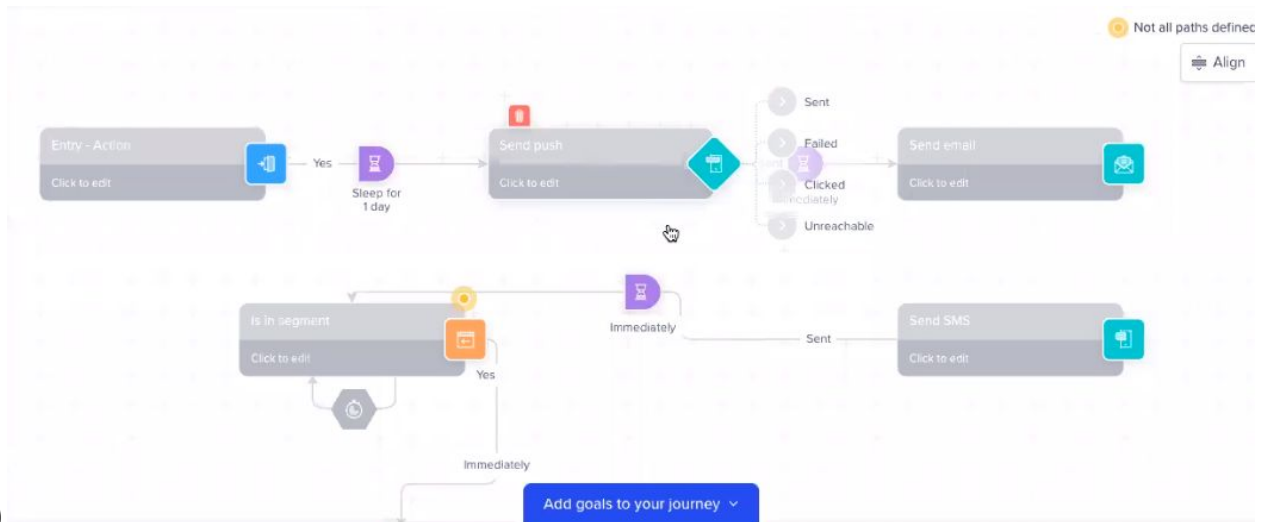
- 42) Recency is on x-axis and frequency on y-axis.
- 43) If users have performed an event quite recently, like today, **they fall in champions segment.**
- 44) **Hibernating** segment states **low recency and low frequency and low monetary impact.**
- 45) **Cannot lose them** segment states these users **have low recency but high frequency and monetary value.** This means that these users were **very active at one point of time on the app but have not interacted with the app quite recently.** These users need to be targeted with campaigns to make them champions again.
- 46) We can click on transitions to see how many users move from one segment to the other. We would want more users becoming champions and not the opposite.
- 47) From Settings, we can turn on session analytics and see how our session performs
- 48) This data is gathered via the "Session Concluded" event when a session ends.
- 49) CT session ID needs to be added to every event.
- 50) This creates a new session dashboard to view
- 51) It takes 20 minutes to raise session concluded events.


3 - Campaigns

- 1) There are primarily two ways to engage with users. a) **Using Campaigns and b) Using Journeys.**
- 2) There are various channels by which we can set up our campaigns. These can be the following:
 - a) **Mobile Channels -**
 - i) **Mobile push**
 - ii) **Mobile In-app**
 - iii) **SMS**
 - iv) **App Inbox**
 - b) **Email Channels**
 - c) **Desktop/Mobile Web Channels**
 - i) **Web Push**
 - ii) **Web Pop Up**
 - iii) **Web Exit Intent**
 - d) **Partner Channels**
 - i) **Amazon EventBridge**
 - ii) **Segment**

- iii) mParticle
 - e) Server Channels
 - i) Webhooks
 - f) Social Channels
 - i) Whatsapp
 - g) Remarketing Channels
 - i) Facebook Audiences
 - ii) Google Ads
- 3) For Remarketing Channels, the customer's google and Fb email id is needed.
 - 4) **Mobile Push** -> Primarily two types of segments -> Past and live behaviour user
 - 5) Past behaviour -> One time and Multiple dates and Recurring campaigns can be created.
 - 6) **Live Users** -> can have inaction with time, single action and on Date/Time
 - 7) In an example of the recurring campaign -> There are different ways to choose the frequency
 - 8) There is a setting called **the best time for every user**, wherein we can send campaigns to **every user individually based on their preferred app usage time**.
 - 9) This can be done via **Settings-> Setup-> Best Time** and selecting the attendance event
 - 10) Primarily there are **global throttle limits** that are applied on an account.
 - 11) These can be set via campaign limits under **settings -> Setup-> Campaign limits** section
 - 12) These are restrictions that **allow us to control campaigns**
 - 13) Throttles mean the total messages sent to all users **every five minutes**. This limit can be used to **control the number of people engaging with the app at a single time**. For example, if an app like Flipkart engages all its users and everyone comes online, it might crash. So we need to **set limits on the number of engagements for very high throughput apps**.
 - 14) There are three ways we can set up our message.
 - a) Single Message
 - b) A/B test- send up to three variants of a message, let's say like A, B, C. Then based on engagement result we can do an A/B test and send the A/B message to the userbase.
 - c) Message on User Property
 - 15) For Android 8+, Notification channels need to be established. These can be done via Settings-> Mobile push
 - 16) We can add channels in **Enable Notification channels for Android 8.0+ and above**.
 - 17) We can use either HTML or pre-existing templates for Emails.

- 18) Mobile-in-app requires the user to be on the app. These particularly use the inbuilt native properties to send a notification within the app.
- 19) App inbox allows mobile push to be sent to inbox within the app. This can be set via clevertap SDK or simply using the API tools if the customer has their own app
- 20) **Journeys** are more complex campaign setups. These typically allow us to chain a lot of campaign events and create a flow which when traversed by user gives us a detailed analysis of the user behaviour while targeting what works best on the user.
- 21) Two Criteria are important in Journeys - **Entry and exit Criteria.**
- 22) It's also important to define the goal. We can define up to 3 goals, like Entry (App Launch)-> Goal(Transaction).
- 23) **Once the user has completed any goal, he/she will be dropped from the Journey.**



- 24)  It is recommended to break up long journeys into smaller ones to track better.
- 26) E.g on a customer onboarding process, if the user has to signup, add a card, KYC etc.all these parts can have their own journeys with a specific goal so as to understand better how these work.

4 - Boards, GeoLocation, Product Experiences
etc.

1. By default, there are 3 boards installed in the cleverTap dashboard.
 - a. **Mobile App**

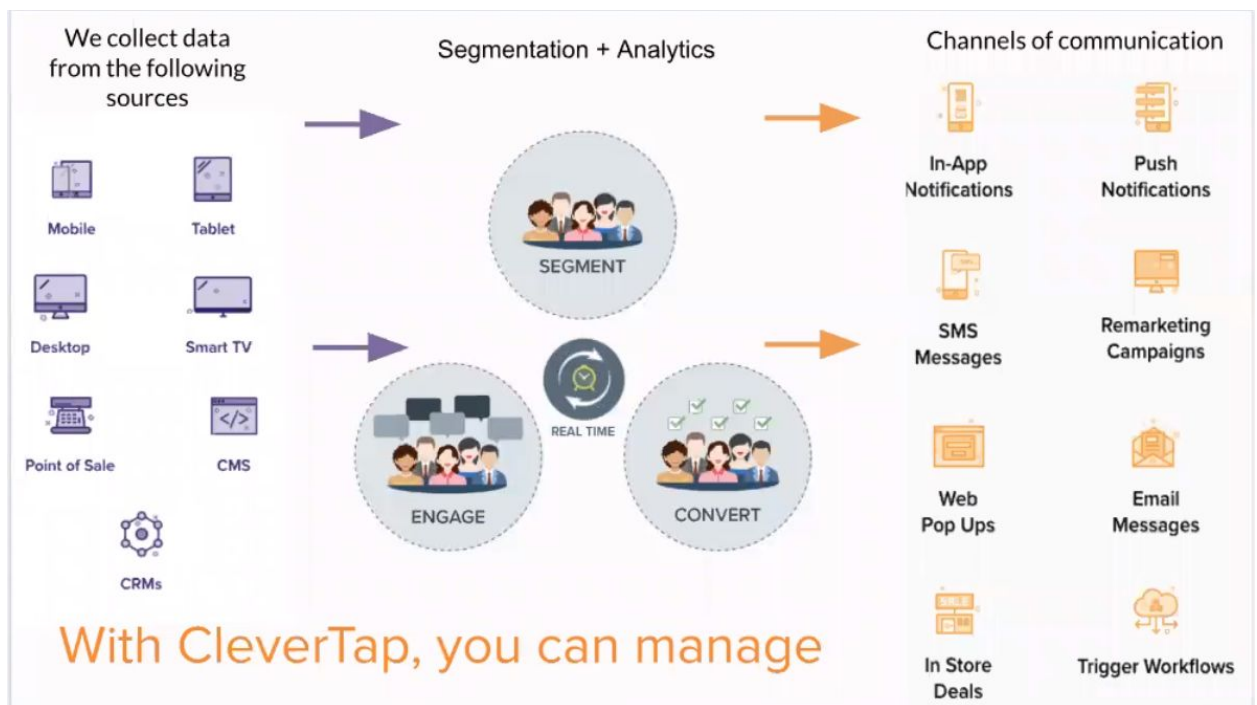
b. Uninstall

c. Revenue

2. Activity Shows events and users interacting with the event
3. **Lifecycle optimizer tracks various stages of users across the customer's lifecycle.**
4. This is done through three key points **Retention Rate, Key Conversion Rate, Repeat Conversion Rate.**
5. There are two types of frameworks that have been provided by CleverTap for lifecycles
 - a. **AIC** - Acknowledgement, Interest, Conversion
 - b. **AARRR**- Aquisition, Activation, Retention, Referral, Revenue.
6. **GeoFencing** - It is simply creating a geographical location cluster where a particular campaign can run.
7. Geofencing can be created by Settings->Engage->Geofences.
8. Longitude and Latitude with reverse IP can be used to track exact locations. But these require user permission.
9. **Bulletins - e.g** if we are purchasing something from amazon and it is not available, we can simply click on a button saying remind me when the item is in stock, whenever then the item comes back in stock, we will get a push notification doing exactly that. A notification about the product will come to us as a bulletin which will remind us about the product. This is the primary functionality.
10. Bulletins can also be used to **provide functionality about a future event** whenever it is possible for it to happen. E.g Tickets of an upcoming movie in BookMyShow being available.
11. Bulletins are **different than campaigns** as:
 - a. If an event needs to be made engageable using campaign, first the event has to occur, then it has to be consumed by cleverTap and then mapped to the campaign.
 - b. This consumes some time and sometimes, like in flash sales, this isn't ideal.
 - c. Here bulletins come into play.
 - d. Without even mapping the event to the user's profile, the event is just taken and sent directly to the user.
 - e. **In a simple sense, rather than sending the event from the app, the customer can create an event on the clevertap dashboard and use it to send a notification to the user.**
12. We can further define segments in Bulletins just like campaigns to track them.
13. To customise the message in push notifications we can use @ to simply specify what user property to use to customize behaviour.

14. **Product A/B test** simply is rolling out different versions to different sets of users to compare and analyse which version performs better and eventually which can be used.
15. There are three parameters used to launch Product A/B tests -
 - a. **Goals** - The end goal for the experiment to be deemed successful.
 - b. **Variants** - The different versions of experiments.
 - c. **Segments** - Groups of users on which the experiments are run upon.
16. Once a variant performs better, then there is the option to roll this out to all users.
17. **Product Experiences** - Product experiences allow the users to unlock certain part and features of an app, based on criteria such as if they have paid for that feature or part.
18. Keys can be set for developers. Whenever Clevertap sends these keys to the user's app, the app can then process these keys on their backend and proceed to unlock the product experiences for the user.
19. **Product config** can be used to select a segment and send campaigns to our target group of users.

4 - SDKs



- 1)
- 2) SDK's play a role in connecting our sources to our platform.
- 3) For every user profile, two types of data are shared, 1) **Who is my user?** (User id, parameters and details) 2) **What is my user doing (Events)?**

- 4) Equivalent methods of these are both present in the SDKs.
- 5) System and Custom user properties are set using these.
- 6) When we are not using identity or email, always use profile.push.
- 7) When the user is logging in or registering, we use the method onUserLogin.
- 8) clevertap.profile.push({"Site":{}}) can be used to push a user profile.
- 9) Clevertap id is automatically generated by clevertapID. This is basically to track anonymous behaviour when the user hasn't signed up.
- 10) We can get it using clevertap.getCleverTapID()
- 11) When a new user logs in and if we want to track them, we need to use clevertap.onUserLogin.push({"Site":{}})
- 12) Basically when we want the user to say log in, then we need to pass identity or email. This should always be tracked with onUserLogin.
- 13) Otherwise, we must use profile.push to track an existing profile.
- 14) For web at verbose level, we can have sessionStorage["WZRK_D"]=""; to start getting logs.
- 15) Merging scenario can come in for example if a user logs in and has a clevertap Id as _g123|1234
 - a) The identity lets suppose to be as Ankush
 - b) Now user uninstalls the app
 - c) New user B logs in with same clevertap id as _g123|1234
 - d) But Identity is different as - Ank
 - e) Now these both users will be tracked under the same user profile as clevertap id is the same
 - f) The solution is to stop using addid for ctid or
 - i) Tell that users won't actually show the behaviour.
 - ii) What this simply means is that users b will not show up as in real-world scenario, usually, the same user simply shows up again.
 - g) Sometimes users on logout clear cache.
 - i) This keeps increasing the number of CTID in the profile
 - ii) This won't introduce a new identity but ctid will increase.
 - iii) We have a limit of 500 ctid per profile.
 - iv) Here the solution is to ask to not remove any ctid on cache clear
 - v) Or tell the customer to clear app cache but not clevertap one
- 16) In **SDK the dependency is introduced via a gradle.**
- 17) We need the following in module level of gradle file
 - a) `//Clevertap`
 - b) `implementation 'com.clevertap.android:clevertap-android-sdk:4.0.0'`
 - c) `implementation 'com.google.firebase:firebase-messaging:20.3.0'`
 - d) `implementation 'com.google.android.gms:play-services-base:17.4.0'`
 - e) `//Mandatory for CleverTap Android SDK v3.6.4 and above add the following -`
 - f) `implementation 'com.android.installreferrer:installreferrer:2.1'`

- g)
- h) `implementation platform('com.google.firebase:firebase-bom:25.12.0')`
- i)
- j) `implementation 'com.google.firebase:firebase-analytics'`
- k) `implementation 'androidx.fragment:fragment:1.1.0'`
- 18) Finally apply plugin at end
 - a) `apply plugin: 'com.google.gms.google-services'`
- 19) The XML file must look like as follows
 - a) `<?xml version="1.0" encoding="utf-8"?>`
 - b) `<manifest xmlns:android="http://schemas.android.com/apk/res/android"`
 - c) `package="com.example.clevertapapp">`
 - d)
 - e) `<!-- Required to allow the app to send events and user profile information -->`
 - f) `<uses-permission android:name="android.permission.INTERNET"/>`
 - g) `<!-- Recommended so that CleverTap knows when to attempt a network call -->`
 - h) `<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>`
 - i)
 - j) `<application`
 - k) `android:name="com.clevertap.android.sdk.Application"`
 - l) `android:allowBackup="true"`
 - m) `android:icon="@mipmap/ic_launcher"`
 - n) `android:label="@string/app_name"`
 - o) `android:roundIcon="@mipmap/ic_launcher_round"`
 - p) `android:supportRtl="true"`
 - q) `android:theme="@style/AppTheme">`
 - r)
 - s) `<meta-data`
 - t) `android:name="CLEVERTAP_ACCOUNT_ID"`
 - u) `android:value="RK4-R6K-565Z"/>`
 - v) `<meta-data`
 - w) `android:name="CLEVERTAP_TOKEN"`
 - x) `android:value="362-432"/>`
 - y)
 - z) `<meta-data`
 - aa) `android:name="FCM_SENDER_ID"`
 - bb) `android:value="id: 36500583758"/>`
 - cc) `<meta-data`
 - dd) `android:name="CLEVERTAP_BACKGROUND_SYNC"`
 - ee) `android:value="1"/>`
 - ff)

gg) `<service android:name=".MyFirebaseMessagingService">`
 hh) `<intent-filter>`
 ii) `<action android:name="com.google.firebase.MESSAGING_EVENT"/>`
 jj) `</intent-filter>`
 kk) `</service>`
 ll)
 mm) `<activity android:name=".MainActivity">`
 nn) `<intent-filter>`
 oo) `<action android:name="android.intent.action.MAIN" />`
 pp)
 qq) `<category android:name="android.intent.category.LAUNCHER" />`
 rr) `</intent-filter>`
 ss) `</activity>`
 tt) `</application>`
 uu)
 vv) `</manifest>`

20) We would want the clevertap sdk to also listen to app

a) `android:name="com.clevertap.android.sdk.Application"`

b) This essentially does that task for us

21) 99% time the apps will have an application class

22) Before `super.onCreate()`, if we add `ActivityLifecycleCallback.register(this);`

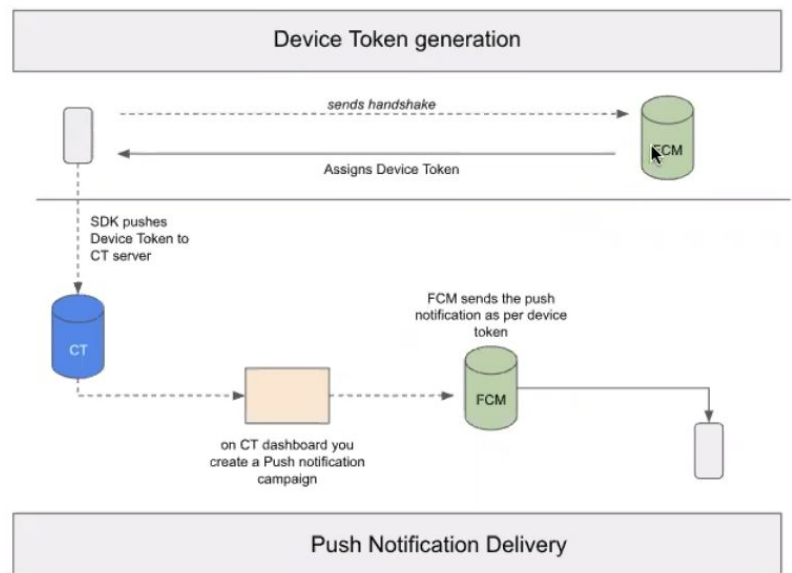
23) `CleverTapAPI.setDebugLevel(3);` This can be used to debug app!

24) For Android 8 and above, channels need to be created

25) So token generation is essentially when you launch the app for the first time, right, a handshake request goes to Firebase.

26)

Explanation



5 - APIs and Webhooks

- 1) Customers do like updating user properties via API right so imagine their internal database having some intelligence or their market research populates pushes data into clever tap, which then we can use for segmentation and of course, there are transactional data with the things which happen outside of the app environment that can be pushed to clevertap via APIs as well.
- 2) Things that can be done via API
 - a) Download user info.
 - b) Upload user info
 - c) Upload push notification tokens
 - i) E.g if we have tokens in the database, we can directly upload everything to clevertap.
 - d) We have also get profile **count API which is equivalent to find people.**
 - e) **Delete user profile API** that complies with GDPR to delete everything on user profile if they don't want to be tracked.
 - f) Download event data
 - g) Upload event data
 - h) Create a campaign via API, this helps in automation.
 - i) Get campaign reports

- j) Get campaign info
 - k) Stop the campaign.
 - l) Message reports
 - m) Real-time counts
 - n) Property counts.
- 3) Up to a thousand JSON, array objects can be sent in the API body.
 - 4) 3 headers need to be configured while sending requests via API post method.
 - 5) If one object has an error, the following JSON is received.
 - a) {


```

              "status": "partial",
              "processed":1,
              "unprocessed":[]
            }
```
 - 6) Tokens are at the device level and not profile level
 - 7) They can be uploaded only against clevertap id and not identity.
 - 8) Count API follow a cursor and a request Id structure
 - a) In a single fetch, we can fetch up to 5000 API record.
 - b) There is a concurrency limit of 3.
 - c) For upload there is a soft limit of 3X1000, that is 3 concurrent 1000 uploads.
 - 9) The only thing not in APIs are journeys, cohorts, segments etc.
 - 10) There is a postman collection that helps in loading everything into the clevertap api.
 - 11) WEBHOOKS are basically the reverse of APIs.**
 - 12) Clevertap pushes data to the customer endpoint.
 - 13) APIs do not allow us to create live user behaviour campaigns, but only past behaviour.

Webhooks allow both.
 - 14) Web hooks can be created in a similar way to the campaign and we need to select the webhook option when creating it via the campaign tab.
 - 15) The workflow of webhooks is to simply create an endpoint, assign headers, put the http method as get or post and create webhook. This will show up in the settings, engage, webhooks tab.
 - 16) Once created the webhook will send a sample payload to the endpoint. If the response returned is 200 ok, then the webhook will show up on the dashboard.
 - 17) There are two payload formats. One is the standard format and the other is the custom body where we can modify the payload according to customer's api or endpoint.